VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 Filip Hudák

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

NOTIFIKAČNÍ ROBOT
NOTIFICATION ROBOT

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Filip Hudák**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Martin Holík**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Filip Hudák ***ID:*** 174312
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Notifikační robot

POKYNY PRO VYPRACOVÁNÍ:

Prozkoumejte možnosti notifikací uživatele na základě událostí systému. Pro notifikace zvolte vhodný komunikační kanál (WhatsApp, Slack, Mattermost, Viber, MS Teams), který bude umožňovat odesílat zprávy na zvolené programové rozhraní. V rámci bakalářské práce proveďte implementaci robota. Zdrojový kód robota by měl být použitelný jako interní programový modul jazyka Python, ale zároveň by měl obsahovat o samostatné externí řešení ovladatelné prostřednictvím Vámi navrženého programového rozhraní. Navržený robot by měl umět kromě odesílání zpráv i reagovat na přijaté zprávy od uživatele například spustěním další události. Další požadovanou funkcionalitou je možnost sledovat novinky na externích webech (například TAČR, GAČR, MVČR, apod.). Součástí robota by mělo být zabezpečené aplikační rozhraní pro připojení aplikací třetích stran s vytvořeným postupem k této implementaci (formou GUI/CMD) na zvolené programové rozhraní.

DOPORUČENÁ LITERATURA:

[1] PUŽMANOVÁ, R. Moderní komunikační sítě od A do Z: [technologie pro datovou, hlasovou i multimediální komunikaci]. 2., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 8025112780.

[2] Pilgrim, M. Ponořme se do Python(u) 3. CZ.NIC, z.s.p.o., 2010, ISBN: 978-80-904248-2-1

Termín zadání: 1.2.2021 ***Termín odevzdání:*** 31.5.2021

Vedoucí práce: Ing. Martin Holík

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Bakalárska práca sa venuje vytvoreniu notifikačného bota na vhodne zvolenom komunikačnom kanály. V teoretickej časti sa porovnávajú rôzne komunikačné kanály s ohľadom na cenovú stránku, súkromie, šifrovanie, popularitu a ďalšie. Praktická časť popisuje registráciu a aplikačné nastavenie komunikácie, testy získavania dát a posielania správ na daný komunikačný kanál. Zaoberá sa návrhom komunikačného modulu s ukladaním tokenu a tajného kľúča spolu s zjednodušeným posielaním správ. Uvádza návrh a štruktúru notifikačného bota, ako aj návrh komunikačných rozhraní s možným pripojením aplikácií tretích strán. Kladie dôraz na rýchle a bezpečné šifrovanie cez vytvorené komunikačné rozhrania pomocou osvedčených postupov. Ďalším bodom v praktickej časti sa opisuje monitorovanie jednotlivých webových stránok s jednoduchým pridávaním webových stránok. Na záver sa preberá inštalácia obrazov a následné nasadenie programov vo vlastných kontajneroch pomocou programu Docker.

KĽÚČOVÉ SLOVÁ

asymetrické šifrovanie, automatizácia, bot, monitorovanie webových stránok, python, python modul, REST API, slack, slackclient, Slack API, symetrické šifrovanie

ABSTRACT

The bachelor thesis focuses on the creation of notification bot for appropriate communication channel. In theoretical part, different communication channels are compared based on price, privacy, encryption, popularity etc. Practical part describes registration and application setting of communication, tests of data acquisition and sending messages to the communication channel. It deals with the design of a communication module, which stores a token and secret key, with simplified method for sending messages. It states design and a structure of the notification bot, as well as the design of communication interfaces for the communication with third party applications. It places importance on fast and secure encryption via created communication interfaces using best practices. Next point in the pratical part describes tracking individual web pages with easy addition of web pages. In conclusion it describes Docker images installation, which is followed by own containers deployment.

KEYWORDS

asymmetric encryption, automatization, bot, python, python module, REST API, slack, slackclient, Slack API, symmetric encryption, web scrapers

Vysadené pomocou balíčku `thesis` verzie 4.03; `http://latex.feec.vutbr.cz`

HUDÁK, Filip. *Notifikačný robot*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2021, 55 s. Bakalárska práca. Vedúci práce: Ing. Martin Holík

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Filip Hudák
VUT ID autora:	174312
Typ práce:	Bakalárska práca
Akademický rok:	2020/21
Téma závěrečnej práce:	Notifikačný robot

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce pánovi Ing. Martinovi Holíkovi za odborné vedenie, konzultácie, trpezlivosť a inšpirujúce návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov

Úvod

Automatizácia je dôležitou súčasťou priemyslu už niekoľko rokov. Je výrazne lacnejšia, efektívnejšia a spoľahlivejšia než ľudská pracovná sila. Svoje uplatnenie nachádza nie len v priemysle ale aj v ostatných oboroch napr. v informatike, zdravotníctve, ekonomike, administratíve a mnoho ďalších. V odvetví informatiky sa automatizáciou myslí sada inštrukcií, akcií a reakcí, ktorá nahradzuje manuálne opakovanú prácu. Čo má za následok dodanie rýchlejšieho, lepšieho a lacnejšieho výsledku.

Cieľom tejto práce je vytvoriť komunikačný modul v programovacom jazyku Python, pomocou ktorého bude možné posielať správy na predom zvolenú komunikačnú platformu. Implementovať bota, ktorý bude posielať a reagovať na správy pomocou predom vytvoreného komunikačného modulu. Pridať botovi funkcionalitu na sledovanie zmeny obsahu webových stránok s nasledovným oznámením na predom zvolenej komunikačnej platforme. Umožniť užívateľom byť notifikovaný nie len v spoločných skupinách (kanálov) ale aj v súkromných správach. Bot má taktiež obsahovať zabezpečené komunikačné rozhranie na komunikáciu s aplikáciami tretích strán.

1 Teoretická časť práce

1.1 Porovnánie komunikačných kanálov

1.1.1 WhatsApp

Základné informácie

WhatsApp, alebo celým menom WhatsApp Messenger je americká multi-platformová freeware chatovacia aplikácia. Táto aplikácia umožňuje užívateľom výmenu súkromných správ, skupinových správ, hlasových správ, obrázkov, videí a súborov. Od roku 2014 túto aplikáciu zastrešuje Facebooku ako svojú dcérsku spoločnosť. Na rozdiel od mena a hesla sa používateľ prihlasuje telefónnym číslom, na ktoré mu bude odoslaný autorizačný kód, čím sa mu mobilný telefón overí. Po vytvorení účtu sa automaticky pridajú kontakty do aplikácie porovnávaním telefónnych kontaktov voči svojej vlastnej databáze. Ako šifrovanie medzi koncovými zariadeniami používa Signal Protocol.[1]

WhatsApp Business API

WhatsApp Business API klient ponúka všetky dostupné funkcie, ktoré sú obsiahnuté v aplikácií WhatsApp pre Android, iOS a web s tým rozdielom, že táto verzia môže byť nasadená na server a poskytovať lokálne *Application Programming Interface – programové rozhranie aplikácií* (API), pomocou ktorého dokážeme prijímať a odosielať správy. Spoločnosť je partnerom mnohých spoločností tretích strán, ktoré dokážu uľahčiť nastavenie a integráciu systému[2].

Po zaregistrovaní v mene firmy a nasledovnej dokončenej verifikácie zo strany spoločnosti Facebook je možné vytvoriť podnikateľský WhatsApp účet.[3] Tento účet okrem iného umožňuje posielať bezplatne vopred nadefinované šablonové správy. Cena za jednu správu sa odvíja od celkového počtu odoslaných správ. V prípade ak obdržíme správu prvý, tak do 24 hodín môžeme v danom chate posielať správy bez poplatku.[4] Cenník správ aplikácie WhatsApp je znázornený v tabuľke Tab.1.1.

Výhody a nevýhody

Medzi najväčšie výhody patrí popularita chatovacej aplikácie a prístupnosť z rôznych zariadení.[5] Svoje využite skôr uplatní ako ChatBot[1] než ako notifikačný bot. Ľahká implementácia pomocou partnerských aplikácií tretích strán. Cena je najväčšou nevýhodou keďže chceme častejšie byť notifikovaný o zmenách, pričom nie sme

[1] program, ktorý simuluje ľudskú konverzáciu

Prvých 250k správ	€ 0.0553
Ďalších 750k správ	€ € 0.0541
Ďalších 2M správ	€ 0.0530
Ďalších 4M správ	€ 0.0496
Ďalších 3M správ	€ 0.0461
Ďalších 5M správ	€ 0.0427
Ďalších 10M správ	€ 0.0427
> 25M správ	€ 0.0427

Tab. 1.1: WhatsApp Business API cenník odoslaných správ

limitovaný len jedným zariadením a na druhej strane je súkromie kde aplikácia si ukladá kontakty na svojom serveri.

1.1.2 Slack

Základné informácie

Slack je vyvíjaný americkou firmou Slack Technologies orientovaný hlavne pre podnikové účely. Ponúka možnosť komunikácie pomocou súkromných správ medzi dvoma užívateľmi. Uľahčuje komunikáciu tímov prostredníctvom komunikačných kanálov[2] a súkromných skupín[6]. Do Slacku je možno integrovať služby tretích strán ako napríklad G Suit, GitHub, Office 365 a ďalšie. Šifruje sa len komunikácia medzi zariadeniami.[7] Šifrovanie dát bolo prenechané k správe administrátorom v spoločnostiach.[8]

Výhody a nevýhody

Populárny nástroj na komunikáciu v spoločnostiach. Možnosť prepojenia s *Active Directory Federation Services – softvérový komponent poskytujúci používateľom SSO* (ADFS) pre riešenie *Single sign–on – systém jednotného prihlásenia* (SSO)[9]. Rozsiahle zdokumentovaná API dokumentácia. Integrácia G Suite a Office 365 v bezplatnom balíku ale na druhú stranu hlasové hovory a videohovory v skupinách sú spoplatnené.[10]

[2]komunikačné miestnosti umožňujúce posielanie textových a hlasových správ, súborov a zahájenie skupinových hovorov a videohovorov v určitej skupine ľudí

1.1.3 Mattermost

Základné informácie

Mattermost je licencovaný ako open-source s možnosťou vlastného server hostingu. Umožňuje posielať instantné správy, obrázky, videá, hudbu a súbory ako také. Aplikácia je dostupná na platformách Android, Windows, *GNU's Not Unix – GNU nie je Unix* (GNU)/Linux, iOS a macOS. Slúži ako Open-Source náhrada pre Slack.[11] Dáta sú pri vlastnom hostingu uložené v lokálnej MySQL alebo PostgreSQL databáze.[12] Zameriava sa hlavne na spoločnosti kde tímy môžu komunikovať v rôznych skupinových kanáloch.

Výhody a nevýhody

Medzi výhodami je to, že aplikácia je open-source, poskytuje vlastný hosting oproti ostatným aplikáciám, možnosť správy užívateľov prostredníctvom služieb Active Directory a *Lightweight Directory Access Protocol – protokol ľahkého prístupu k adresáru* (LDAP) a mnoho ďalších.[12] Nevýhodou je, že aj pri vlastnom hostení aplikácie sú bezplatne dostupné len bežné funkcie aplikácie.[13]

1.1.4 Viber

Základné informácie

Rakuten Viber je japonská freeware multi-platformová servisná služba poskytujúca IP telefóniu, možnosť posielanie okamžitých správ a Viber Out, čo je platená medzinárodná linka s možnosťou volania na mobilné zariadenia.[14] Aplikácia je dostupná na platformách Android, iOS, Windows, GNU/Linux a macOS. Overenie aplikácie prebieha pomocou telefónneho čísla, na ktorý mu bude poslaný autorizačný kód. Viber od verzie 6.0 už začal používať šifrovanie na strane koncových zariadení pomocou vlastného protokolu, ktorý je konceptom Signal protokolu ale nezávislé od jeho autora Open Whisper Systems.[15]

Výhody a nevýhody

Existujúci modul pre Python na komunikáciu s API. Nie najpopulárnejšia ale stále populárna aplikácia.[16]

1.1.5 Telegram

Základné informácie

Telegram je ruská multi-platformová cloudová služba na posielanie okamžitých správ, obrázkov, videí s podporou video a IP telefónie, hudby, obrázkov a dokumentov do veľkosti 2GB.[17] Klientske aplikácie sú dostupné pre Android, iOS, Windows, GNU / LINUX a macOS, ktoré sú licencované pod záštitou GNU GPL v. 2.[18] Hovory sú šifrované na koncových zariadeniach ako aj voliteľný tajný chat, kde na druhú stranu normálny chat a skupinové konverzácie používajú klient-server/server-klient šifrovanie.[19]

Výhody a nevýhody

Výhodou je otvorený zdrojový kód aplikácií a vlastný wrapper ako modul pre Python, ktorý je jednoduchý na použitie. Medzi nevýhodami je prístup aplikácie ku kontaktom.

1.1.6 LINE

Základné informácie

LINE je freeware mobilná aplikácia na okamžité posielanie správ juhokórejského pôvodu. Umožňuje taktiež hlasové správy, posielanie obrázkov a súborov. Poskytuje zdarma VOIP telefóniu a videokonferencie. Platforma LINE dokonca poskytuje svoje proprietárne služby ako LINE Pay[3], LINE Today[4], LINE TV[5], LINE Manga a LINE Webtoon[6]. Line Corporation je dcérska spoločnosť juhokórejského developerského giganta Naver Corporation. Správy šifrujú na koncových zariadeniach pomocou Diffieho-Hellmanového protokolu s použitím eliptických kriviek.[20]

Výhody a nevýhody

Line Corporation poskytuje svoje API zdarma, má implementovanú oficiálnu knižnicu pre Python s aktuálnou dokumentáciou. Ako mínus je popularita mobilnej aplikácie v Európe.[21]

[3] digitálna peňaženka
[4] služba, ktorá streamuje novinky z daného kraja
[5] služba poskytujúca videá na požiadanie
[6] služba distribujúca digitálne komixy

2 Praktická časť práce

Prvá časť praktickej časti práce zahŕňa vytvorenie aplikácie v Slack API, zadefinovanie práv tzn. aké udalosti aplikácia bude odoberať, nastavenie *Uniform Resource Locator – označuje zdroj na internete a protokol, ktorým je možno k zdroju pristupovať* (URL), na ktorú Slack API bude posielať odoberané udalosti a na koniec získanie autentizačného tokenu potrebného pre správnu funkčnosť notifikačného bota.

Druhá časť demonštruje použitie Slack Events API kde bot načúva a náležite reaguje na príkazy od užívateľov. Príkazy sú rozdelené na dve skupiny a to na príkazy, ktoré priamo spomínajú bota v danej správe pomocou zavináča a na príkazy začínajúce lomítkom. Bot odpovedá odoslaním správy alebo súboru na prvú skupinu príkazov pokiaľ správa spomína bota a zároveň obsahuje kľúčové slovo, skrz ktoré rozhoduje čo daný príkaz bude vykonávať. Pomocou lomítka sa v tomto prípade jedná o príkazy súvisiace s výpisom a odberom noviniek z monitorovaných webových stránok.

Tretia časť je zameraná na aplikačné rozhranie, pomocou ktorého je možné komunikovať s botom pomocou aplikácií tretích strán. Komunikácia prebieha pomocou *REpresentational State Transfer – architektonický štýl na distribuovanie hypermediálneho obsahu* (REST) API metód pričom správa je symetricky šifrovaná pomocou kľúča, ktorý je nasledovne zašifrovaný pomocou asymetrickej šifry.

Konečná časť sa venuje nasedeniu programov do vlastných kontajnerov pomocou programu Docker.

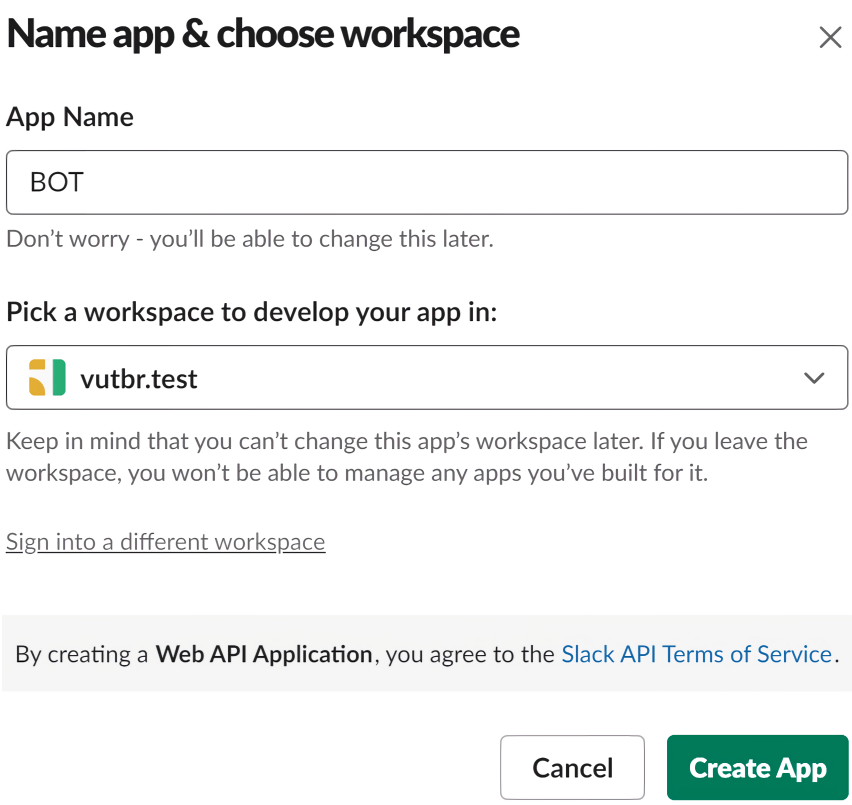
2.1 Slack API

2.1.1 Vytvorenie aplikácie v Slack API

Po prihlásení na stránke Slack API[1] sa nová aplikácia vytvorí pomocou tlačidla `Create New App`, kde sa nasledovne zobrazí formulárové okno so vstupným poľom s názovom aplikácie a rozbaľovacou ponukou, pomocou ktorej sa aplikácia zaradí do daného pracoviska. Viď obrázok Obr.2.1.

Vytvorením aplikácie sa zobrazí nástenka so základným informáciami a nastaveniami. Na to aby aplikácia mohla volať Slack API metódy tak potrebuje mať priradené oprávnenia, ktoré sa nastavia pomocou tlačidla `Permissions` alebo výberom `OAuth & Permissions` z bočného panela.

[1] `https://api.slack.com/`

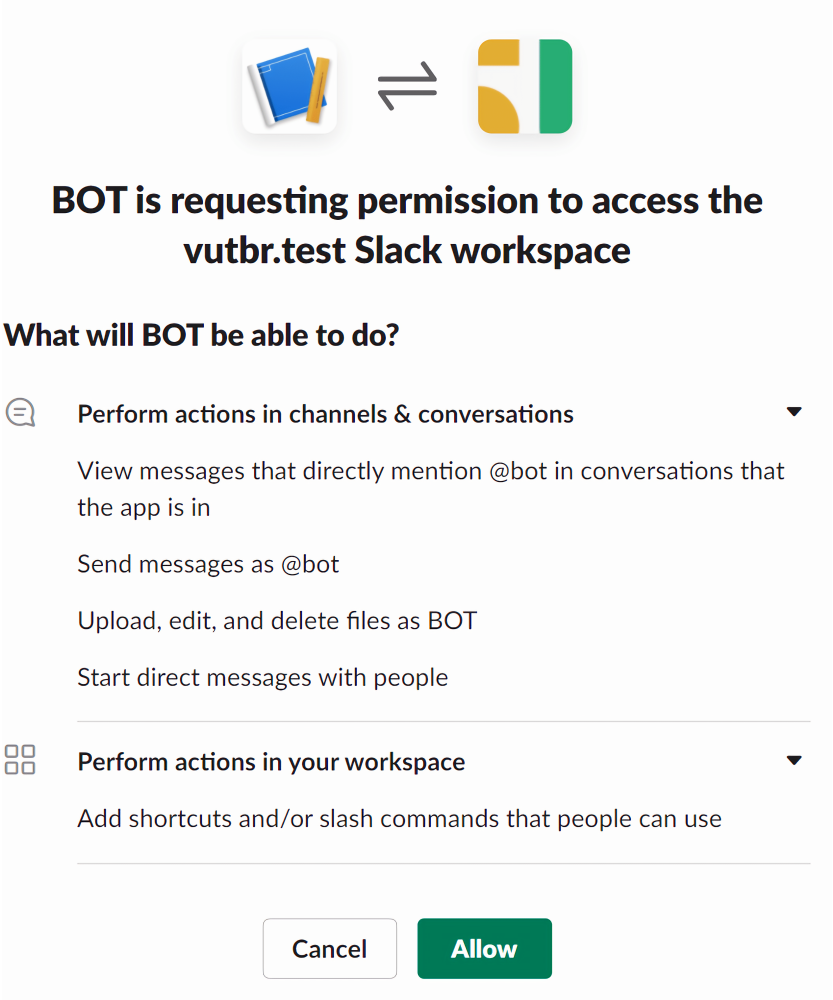


Obr. 2.1: Vytvorenie aplikácie s názvom a priradeným pracoviskom.

V sekcií `Scopes`, presnejšie v podsekcií `Bot Token Scopes` je nutné pridať povolenia pre bežnú komunikáciu s užívateľmi buďto jednotlivo v súkromných správach, alebo hromadne v komunikačných kanáloch pracoviska. Náležité práva, ktoré sa priradia pomocou tlačidla `Add an OAuth Scope`.

- `app_mentions:read` – zobrazenie správ, ktoré priamo spomínajú aplikáciu
- `chat:write` – posielanie správ z aplikácie
- `commands` – povolenie príkazov začínajúcich lomítkom
- `files:write` – posielanie, upravovanie a mazanie súborov z aplikácie
- `im:write` – začatie konverzácie s užívateľmi v súkromných správach

Po nastavení práv sa aplikácia nainštaluje do pracoviska pomocou tlačidla `Install to Workspace`. Odkliknutím tlačidla sa zobrazí oznamovacie okno s rekapituláciou povolení aplikácie. Potvrdením inštalácie tlačidom `Allow` sa priradia aplikácií povolenia a vygeneruje sa OAuth Token.



Obr. 2.2: Súhrn povolení aplikácie.

V tomto štádiu je už možné pomocou OAuth Token komunikovať so Slack API. Na to aby bola aplikácia schopná posielať správy do kanálov pracoviska prípadne súkromné správy užívateľom v pracovisku potrebuje byť pridaná do daného pracoviska. Postup je detailne naznačený na obrázku Obr.A.1 v prílohe.

2.1.2 Test volania Slack API metód

Test volania metód Slack API je možné vykonať niekoľkými spôsobmi. Jednou z možností je pomocou nástroja príkazového riadka v GNU/Linux. V tomto prípade je vybraný nástroj z unixového shellu `cURL`[2], ktorý umožňuje prenášanie dát pomocou rôzných sieťových protokolov. Na získanie zoznamu všetkých kanálov a skupín v pracovnom prostredí slúži metóda `conversations.list`, ktorá vracia vo formáte *JavaScript Object Notation – zápis dátového formátu nezávislého na počítačovej platforme* (JSON) zoznam kanálov, do ktorých má daný OAuth Token prístup.[22]

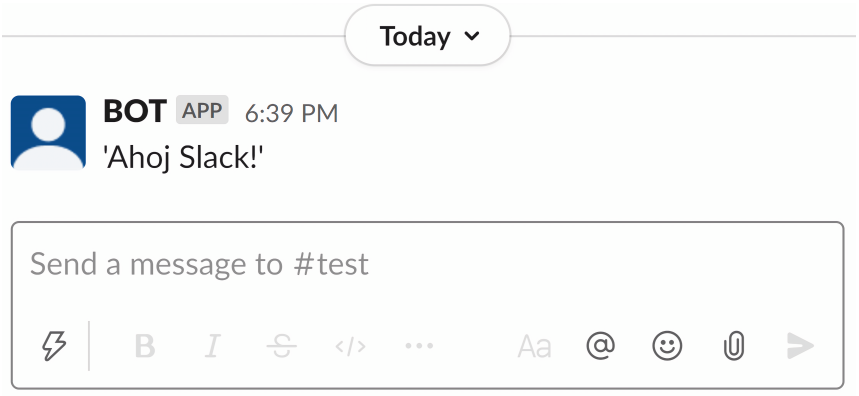
```
curl -s -F 'token='"${my_oauth_token}" \
    "https://slack.com/api/conversations.list" |
    json_pp -f json |
    grep '"ok" :'
```

Nástroj cURL je volaný s možnosťami `-s, --silent`, ktorý skrýva výpisy priebehu a `-F, --form`, ktorý umožňuje poslať vyplňený formulár s parametrom OAuth Tokenu. Výstup je poslaný do `json_pp` kde sa jedná o ďalší nástroj príkazového riadka, ktorý naformátuje výstup z nástroja cURL do čitateľnej podoby. Volá sa s možnosťou `-f`, ktorá značí aký formát má očakávať na vstupe. Na konci príkaz `grep` odfiltruje riadok obsahujúci pár s názvom `ok` a jeho dohnotou. Spustením príkaz vráti `"ok" : true,` čo znamená, že volanie metódy Slack API pomocou OAuth Tokenu dopadlo v poriadku. V prípade poslania správ na kanál sa použije modifikovaný cURL príkaz, ktorý obsahuje pridané parametre možnosti `-F` a to `channel`, ktorým sa špecifikuje kanál, na ktorý sa bude daná správa posielať a `text` s telom správy, ktoré sú potrebné na volanie Slack API metódy `chat.postMessage`[23].

```
curl -s -F 'token='"${my_bot_token}" \
    -F 'channel='"${my_channel_name_or_id}" \
    -F "text='Ahoj Slack!'" \
    "https://slack.com/api/chat.postMessage" |
    json_pp -f json |
    grep '"ok" : '
```

Výstup z príkazu je opäť `"ok" : true,` čo značí, že volanie dopadlo úspešne a správa bola odoslaná. Obrázok Obr.2.3 značí pohľad z kanála úspešného odoslania správy.

[2] `https://curl.se/`



Obr. 2.3: Úspešné odoslanie správy z príkazového riadka.

2.1.3 Nastavenie odberu udalostí

Nato aby bot mohol reagovať na udalosti, tak musí mať aplikácia nastavenú URL, na ktorú bude Slack API posielať odoberané udalosti z aplikácie. To umožní botovi interagovať s užívateľmi v textových kanáloch. Slack Events API pri vykonanej udalosti posiela *Hypertext Transfer Protocol – internetový protokol určený pre komunikáciu s web servermi* (HTTP) *metóda požiadavku HTTP, ktorá žiada webový server aby prijal dáta uložené v tele požiadavku* (POST) žiadosť. Telo žiadosti je formátu JSON, kde je uchovaná informácia o udalosti aká sa vykonala a všeobecné informácie o danej správe ako kto je jej autorom, z akého tímu a kanála bola správa poslaná a podobne[24].

Prihlásenie sa k odberu udalostí

K odberu sa prihlasuje výberom `Event Subscriptions` z hlavného menu kde sa po povolení `Enable Events` zobrazí vstupné pole s možnosťou vyplnenia odoberanej URL. Spojenie sa verifikuje pomocou `url_verification` udalosti[25]. Pre úspešnú verifikáciu musí byť spustený webový server na danej URL adrese, ktorý obdrží verifikačnú žiadosť s telom vo formáte JSON v tvare:

- `token` – zastaralá verzia overenia ale stále používaná
- `challenge` – náhodne vygenerovaný reťazec
- `type` – typ udalosti

Na odlíšenie URL overenia od ostatných udalostí sa používa práve typ `url_verification`.

Žiadosť o overenie URL pomocou výzvy vyzerá nasledovne:

```
{
    "token": "xyzTOKENxyz",
    "challenge": "xyzCHALLENGExyz",
    "type": "url_verification"
}
```

Odpoveď na verifikačnú udalosť nevyžaduje mať nastavené žiadne povolenia. Po obdržaní žiadosti je nutné odpovedať správou s odpoveďou HTTP 200, poľom hlavičky `Content-Type` nastaveným na `text/plain`, `application/x-www-form-urlencoded` alebo `application/json` a s prijatou výzvou (`challenge`) v tele prijatej žiadosti.

```
HTTP 200 OK
Content-type: text/plain
xyzCHALLENGExyz
```

Po úspešnom overení sa nad vstupným poľom zobrazí zelený text `Verified` čo značí úspešné overenie URL a od tohto kroku Slack API začína posielať odoberané udalosti po ich zachytení pomocou HTTP POST žiadostí.

2.1.4 Ngrok

Na testovacie účely bol použitý tunelovací program ngrok[3], ktorý umožňuje vystaviť webový server na lokálnom zariadení aby bol prístupný z internetu. Umožňuje testovanie bota na lokálnom zariadení bez použitia hostovacích služieb tretích strán. Po stiahnutí programu je tunel vytvorený ngrok agentom časovo obmedzený. Na zrušenie časového obmedzenia sa musí ngrok agent overiť. Na jeho overenie sa použije Authtoken, ktorý sa získa vytvorením účtu a následovným prihlásením na domovských stránkach programu[4]. Ngrok agent overí pomocou získaného Authtokenu príkazom:

```
ngrok authtoken <vygenerovaný Authtoken>
```

Nakoniec sa tunel s vlastným číslom portu vytvorí príkazom:

```
ngrok http <port>
```

Obe príkazy sú spustené z príkazového riadka s tým, že program ngrok sa nachádza v aktuálnom pracovnom adresári.
S týmito prerekvizami má bot všetko potrebné na to, aby mohol posielať správy

[3] `https://ngrok.com/`
[4] `https://dashboard.ngrok.com/get-started/your-authtoken`

prípadne reagovať na spomenutia v textových kanáloch pracoviska a odpovedať užívateľom na lomítkové príkazy.

2.2 Komunikačný modul

Návrh komunikačného modulu

Modul `notification_bot` obsahuje triedu `NotificationBot`, ktorá slúži na uloženie OAuth Tokenu, predvoleného kanála a tajného kľúča používaného k overeniu pôvodnosti prijatej POST žiadosti zo Slack API. Predvoleným kanálom sa docieli toho, že bot môže vypisovať konfiguračné správy na jedno miesto v pracovisku zatiaľ čo užívateľom odpovedá na správy do rovnakého kanála z akého posielali správu. Modul taktiež vyhadzuje vlastné výnimky pri chybnom tele URL overovacej žiadosti. Okrem práce s danou triedou, asynchrónne funkcie modulu taktiež rozširujú oficiálny pythonovský modul o funkcie pre jednoduchšiu komunikáciu so Slack API. Zdrojové kódy definície a inicializácie triedy, a následné inicializovanie tokenu s predvoleným kanálom pomocou danej triedy sú zvlášť znázornené v prílohách B.1 a B.2.

Odosielanie správ

Ako bolo vyššie spomínané, na posielanie správy sa používa oficiálny modul, ktorý má v sebe definované základné metódy Slack API ako `chat_postMessage`, ktorá slúži na posielanie správ do kanálov a `files.upload` zase na posielanie súborov[26]. Asynchrónna funkcia `send_message` zaisťuje volanie Slack API metódy `chat_postMessage` zatiaľ čo funkcia `send_file` volá metódu `files.upload`[27]. Vykonávajú sa prostredníctvom úloh, ktoré spúšťajú korutiny v cykloch udalostí. Prijímajú dva parametre typu reťazec. Prvý je text správy prípadne cesta k súboru na odoslanie a druhý, nepovinný, je ID alebo názov kanála v pracovisku. V prípade ak by bol v argumente neplatný kanál alebo neinicializovaný token v triede, tak funkcia ošetrí výnimku `SlackApiError`, ktorú vráti Slack API a nsledovne funkcia vráti chybný návratový kód `-1`. Modul taktiež rieši URL verifikáciu funkciou `authentication`. V argumente funkcia očakáva slovník s telom žiadosti a nasledovne vracia reťazec so správnym formátom odpovede pre úspešné overenie. V prípade chybného tela žiadosti funkcia vráti objekt None.

Výpis 2.1: Príklad posielania správy rozšírením oficiálneho modulu

```
async def task_send_message(m_message: str,
    m_channel: str) -> int:
    try:
        client = slack.WebClient(token=notification_bot.token,
            run_async=True)
        client.chat_postMessage(channel=m_channel,
            text=m_message)
        return 0
    except SlackApiError as e:
        return 1

async def send_message(m_message: str,
    m_channel: str = notification_bot.channel) -> int:

    if m_channel is None:
        return 1
    task = asyncio.create_task(
        task_send_message(m_message, m_channel))
    return await task
```

2.3 Notifikačný bot

Reagovanie na udalosti vykonáva webový server, ktorý v tomto prípade je postavený na webovom frameworku Tornado[5] programovacieho jazyka Python. Hlavný súbor `main.py` pozostáva z `make_app` funkcie kde je celá logika aplikácie, prebieha tam inicializácia dátových tried zo súboru `config.py`, načítanie handlerov z adresára `handlers` spolu so zadefinovaním ich koncových bodov, na ktorých budú načúvať, stanovením sieťového nastavenia a na konci vytvorením asynchrónnej slučky ak bol tento skript spustený ako hlavný program. V prílohe sa nachádzajú časti kódov konfiguračného súboru (B.3), hlavného spúšťacieho skriptu notifikačného bota (B.4) a súboru s handlerom webového servera (B.5).

[5] `https://www.tornadoweb.org/`

Koncové body handlerov

Pre jednoduchšie rozdelenie žiadostí služieb notifikačného bota je každému handleru priradený vlastný koncový bod, na ktorom obsluhuje svoje žiadosti. Notifikačný bot načúva na štyroch koncových bodoch.

- `/robot` – komunikácia so Slack API
- `/slash` – lomítkové príkazy
- `/webscrapers` – prijímanie žiadostí z monitorovaných stránok
- `/slackbot/api` – odpovedanie na žiadosti tretích strán

Kontrola autenticity Slack API žiadosti

`Signing secret` je jedinečný reťazec vygenerovaný pre aplikáciu v kategórií `Basic Information` a sekcií `App Credentials` po prihlásení na hlavných stránkach Slack API. Slack API posiela v každých hlavičkách svojích HTTP žiadostí autentizačný kód, podpis, ktorý je vytvorený hašom tela v kombinácií s tajným šifrovacím kľúčom, teda signing secret[28]. Na vytvorenie hašu sa používa kryptografická hašovacia funkcia `SHA256`. Postup vytvorenia podpisu žiadosti:

- získanie časovej známky z hlavičky z páru s menom `X-Slack-Request-Timestamp`
- vytvorenie reťazca spojením základného reťazca (číslo verzie - momentálne v0), časovej známky a pôvodného, nekonvertovaného, tela oddelenými dvojbodkami (`v0:123456789:command=/weather&text=94070`).
- vytvoriť podpis správy pomocou hašovacej funkcie a tajného kódu
- naformátovanie reťazca s podpisom do správného formátu (v0=xyzPODPISxyz)

Pri porovnaní sa vypočítaný podpis porovná s podpisom v hlavičke žiadosti v páre s názvom `X-Slack-Signature`. Na hašovanie a výpočet podpisu sa používajú pythonovské knižnice `hashlib` a `hmac`.

Výpis 2.2: Príklad kontroly autenticity Slack API žiadosti

```
import hmac
import hashlib


async def slack_api_hash_validator(
    m_headers: tornado.httputil.HTTPHeaders,
    m_body: bytes) -> bool:

    slack_timestamp: str = m_headers['X-Slack-Request-Timestamp']
    slack_signing_secret: str = await nb.get_signing_secret()
    request_body = m_body.decode("UTF-8")

    if abs(time.time() - int(slack_timestamp)) > 60 * 5:
        return False

    sig_basestring = f"v0:{slack_timestamp}:{str(request_body)}"
    my_signature = f"v0={
                          hmac.new(str.encode(slack_signing_secret),
                          str.encode(sig_basestring),
                          hashlib.sha256).hexdigest()
                      }"

    slack_signature = m_headers['X-Slack-Signature']


    return True if (hmac.compare_digest(
                                        my_signature,
                                        slack_signature
                                    )) else False
```

2.3.1 Premenné prostredia

Bot obsahuje nastaviteľné parametre pre chod Tornado servera v premenných prostredia, ktoré sú rozdelené na povinné premenné, ktoré sú dôležité na spustenie notifikačného bota, a na voliteľné premenné, ktoré sú predvolene už preddefinované teda nie sú vyžadované aby boli zadefinované pred štartom notifikačného bota. Obe skupiny premenných prostredia sa inicializujú v dátových triedach `ServerConfig` a `StaticConfig` súboru `config.py`.

Povinné premenné prostredia

- `SLACK_TOKEN` – obsahuje OAuth Token vygenerovaný z hlavných stánok Slack API
- `SLACK_SIGNING_SECRET` – obsahuje signing secret vygenerovaný z hlavných stánok Slack API

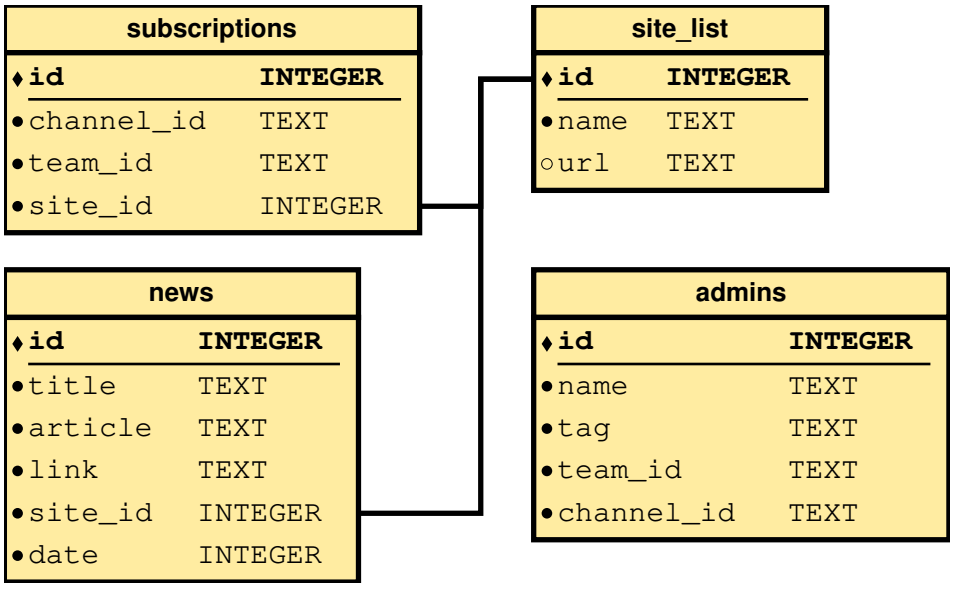
Voliteľné premenné prostredia

- `TORNADO_SERVER_PORT` – port, na ktorom naslúcha notifikačný bot [4390]
- `TORNADO_IP_ADDRESS` – IP adresa, na ktorej beží notifikačný bot [127.0.0.1]
- `IMGS_DIR` – obsahuje cestu k adresáru s obrázkami [images]
- `PRIVATE_KEY_FILEPATH` – obsahuje cestu k súboru s privátnym kľúčom notifikačného bota [certs/main/private_key.pem]
- `PRIVATE_KEY_PASSWORD` – heslo, ktorým bol zašifrovaný privátny kľúč [None]
- `CERTS_DIR` – obsahuje cestu k adresáru s verejnými kľúčmi [certs]

2.3.2 Ukladanie dát

Notifikačný bot používa k uloženiu dát knižnicu SQLite. Jedná sa o embedovaný SQL databázový engine, ktorý na rozdiel od ostatných databázových enginov nepracuje na báze klient-server ale číta a zapisuje dáta priamo na disk do súboru[29]. Databáze, s ktorými pracuje notifikačný bot sú uložené v adresári `databases`. V prvej databáze `main.db` sa nachádzajú tabuľky:

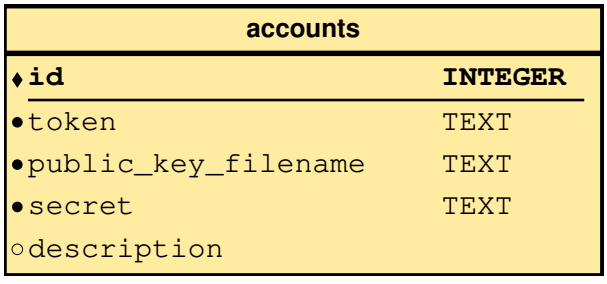
- `admins` – zoznam užívateľov oprávnených k používaniu `set` príkazov
- `news` – zoznam noviniek za posledných 30 dní z monitorovaných webových stránok
- `site_list` – zoznam monitorovaných stránok
- `subscriptions` – zoznam kanálov/užívateľov s odoberanými stránkami



Obr. 2.4: Diagram databáze `main.db`

Nato aby aplikácie tretích strán mohli komunikovať s notifikačným botom, tak potrebujú byť pridané do databáze v súbore `auth.db`. Databáza obsahuje tabuľku `accounts`, do ktorej sa ukladajú záznamy `token`, `public`, `token`, `token`,

- `token` – vygenerovaný token, pomocou ktorého je možno jednoznačne identifikovať aplikáciu
- `public_key_filename` – názov lokálne uloženého verejného kľúča aplikácie k zašifrovaniu správy posielanej cez API
- `secret` – tajný kľúč používaný k vytvoreniu alebo overeniu podpisu pôvodnosti žiadosti
- `description` – popis



Obr. 2.5: Diagram databáze `auth.db`

2.3.3 Operátor databáze

Trieda `My_Database_Operator` je zjednodušením práce nad databázami SQLite a je definovaná v súbore `database_connector_class.py`. Do triedy sa ukladajú dve premenné s cestou k súboru databáze a inštancia kurzora databáze. Cesta k súboru sa nastaví v parametre konštruktora triedy a kurzor sa nastaví sám po pripojení k databáze pomocou metódy `connect`. Metódy `create`, `select_one`, `select_all`, `insert`, `update` a `delete` prijímajú dva parametre. Prvý je reťazec so samotným SQL príkazom a v druhom, nepovinnom, metóda očakáva n-ticu, ktorej hodnoty kurzor pri volaní SQLite metód nahradí namiesto znaku otáznika v SQL príkaze z prvého parametra. Vykonané zmeny v databáze sa uložia metódou `commit` a nakoniec ukončenie spojenia s databázou zakončí metóda `close`.

2.3.4 Logovanie

Notifikačný bot loguje priamo na výstup do konzole správy, ktoré majú level dôležitosti INFO a vyšší. Správy všetkých levelov taktiež ukladá do súboru `logs/server_log.log`. Samotná konfigurácia logera notifikačného bota sa nachádza v súbore `server_logger.py`.

2.3.5 Obsluha príkazov spomenutím robota

Pokiaľ správa obsahuje spomenutie robota, názvu bota so zavináčom ako prefix, tak sa jedná o príkazy spomenutím a rieši ich hlandler `slack_bot_handler.py` v adresári `handlers`. Po kontrole formátu a metódy žiadosti sa nasledovne kontroluje či sedí vypočítaný podpis s podpisom v hlavičke žiadosti. Po úspešnom overení pravosti sa z tela získa text správy, ktorý je uložený pod kľúčom `event` a podkľúčom `text`. V danej správe sa ďalej hľadajú jednotlivé podreťazce a v prípade výskytu sa vykoná patričný príkaz, inak v opačnom prípade ak sa nevyskytuje žiaden podreťazec v správy tak bot vypíše, že sa jedná o neplatný príkaz. Pre príklad boli vytvorené príkazy `ping`, `vut`, `set channel` a `help`. Na príkaz `ping` odpovie bot správou `pong`. Príklad poslania obrázku sa vykoná pomocou príkazu `vut`. Ak užívateľ nemá dostatočné oprávenie, nenachádza sa v tabuľke `admins`, tak ho bot nepustí nastaviť predvolený kanál na odosielanie správ a na koniec príkaz `help` vraví sám za seba. Odpovie so zoznamom podporovaných príkazov užívateľovi.

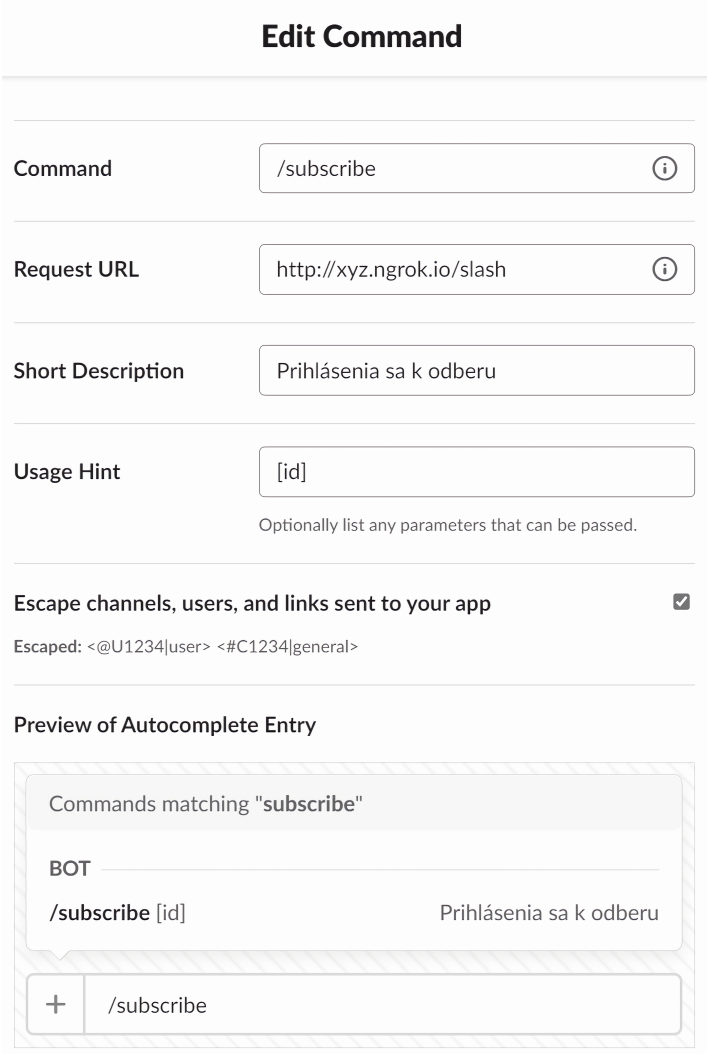
2.3.6 Obsluha lomítkových príkazov

Ak správa začína lomítkom tak Slack aplikácia automaticky napovie užívateľovi doplňovaním aké lomítkové príkazy môže použiť. O lomítkové príkazy sa stará handler `slash_commands_handler.py`. Pred spracovaním žiadosti sa kontroluje HTTP

metóda, formát a nakoniec podpis pôvodnosti tela. V prípade ak dopadne nejaká z kontrol neúspešne, žiadosť sa zahadzuje. V opačnom prípade sa žiadosť spracováva v hlavnej metóde triedy.

Definícia jednotlivých príkazov v Slack API

Na vytvorenie nového lomítkového príkazu sa najprv príkaz potrebuje zadefinovať na hlavných stránkach Slack API výberom kategórie `Slash Commands` z bočného panela. Po kliknutí na tlačidlo `Create New Command` sa zobrazí okno s možnými nastaveniami príkazu ako názov príkazu, URL, na ktorú sa bude posielať HTTP žiadosť zo Slack API, popis príkazu a napovedá k použitiu.



Obr. 2.6: Príklad vytvorenia lomítkového príkazu.

Realizácia príkazov

Lomítkové príkazy spravuje handler `slash_commands_handler.py`. Podporované príkazy sú `list`, `subscribe`, `unsubscribe`, `subscriptions`, `getnews` a `help`. Výpis monitorovaných stránok s ich identifikačnými číslami sa docieli príkazom `list`, ktorý odpovie s načítanými záznamy z tabuľky `site_list`. Príkazy `subscribe`, `unsubscribe` a `subscriptions` sa starajú ako už z názvu vyplýva o odbery užívateľov a kanálov. Dopĺňajú, mažú alebo vypisujú záznamy z tabuľky `subscriptions`. Príkazy `subscribe` a `unsubscribe` majú povinný argument, ktorý obsahuje identifikačné číslo monitorovanej stránky pomocou, ktorého nasledovne pridajú alebo odoberú svoj odber. Argumentom príkazu `getnews` je možné určiť počet vypísaných noviniek danej stránky. Príkaz má povinný argument identifikačného čísla stránky a nepovinný s počtom vypísaných noviniek. Bez udania počtu výpisov sa vypíše jeden záznam.

2.3.7 Žiadosti z monitorovaných webových stránok

Notifikačný bot má pre žiadosti od monitorovaných stránok vyhradený handler `web_scraper_api_handler.py`. Po prijatí žiadosti a jej spracovaní na ňu odpovedá len návratovým kódom s hodnotou `200` prípadne `500` ak nastala chyba pri spracovaní žiadostí. HTTP metóda žiadostí musí byť POST, v opačnom prípade bude žiadosť zahodená.

2.3.8 Prijímanie a výpis noviniek

Potom sa začne kontrolovať či telo žiadosti je vo formáte JSON a obsahuje kľúče `title`, `article`, `link`, a `site_id`. Taktiež sa kontroluje aj hlavička či pozostáva z `timestamp`, `token` a `signing_secret`. Po úspešných kontrolách hlavičiek a tela žiadosti sa na koniec vypočíta podpis a skontroluje či sedí s podpisom v hlavičke žiadosti. Po podarenom overení dôveryhodnosti žiadosti sa pomocou `site_id` získa z databáze zoznam kanálov odoberajúcich danú stránku a následne sa rozpošlú správy pomocou modulu `notification_bot`.

Podpis žiadosti

K tomu aby bot mohol vytvoriť podpis z tela žiadosti potrebuje tajný kľúč, signing secret, ktorý získa z databáze `auth.db` pomocou jedinečného tokenu z hlavičky žiadosti. Podpis sa vytvorí z tela spojením základného reťazca VUT oddeleného medzerou s časovou známkou a telom správy, ktoré sú oddelené dvojbodkou. Vytvorený podpis sa znovu spojí so základným reťazcom a porovná a s hlavičkou žiadosti. Príklad reťazca pred vytvorením podpisu: `VUT 123456:{"body": "telo"}`.

Výpis 2.3: Kontrola autenticity žiadosti z monitorovaných stránok

```
import hmac
import hashlib
import notification_bot as nb

async def check_hash(
    m_payload: dict,
    m_timestamp: str,
    m_token: str,
    m_signature: str
) -> bool:
    if (int(time.time()) - int(m_timestamp)) >= 5 * 60:
        return False

    auth_db: My_Database_Operator = My_Database_Operator(
                                        'databases/auth.db'
                                    )
    auth_db.connect()

    sql: str = f"SELECT secret FROM accounts WHERE token = ?;"
    val: tuple = (m_token,)

    secret: str = None
    try:
        secret = auth_db.select_one(sql, val)[0]
        auth_db.close()
    except MyDatabaseOperatorException as e:
        return False
    except TypeError as e:
        return False

    sign_base_string: str = f"VUT {str(m_timestamp)}:{str(m_payload)}"
    signed_payload: str = f"VUT {hmac.new(str.encode(secret),
                                str.encode(sign_base_string),
                                hashlib.sha256).hexdigest()}"

    if not (hmac.compare_digest(signed_payload, m_signature)):
        return False

    return True
```

2.3.9 Aplikačné rozhranie

Komunikácia s aplikáciami tretích strán sa realizuje pomocou REST API metód. Notifikačný bot realizuje štyri metódy:

- `GET` – získanie záznamov z databáze
- `POST` – vkladanie záznamov do databázy a posielanie správ cez Slack API
- `UPDATE` – aktualizovanie záznamov v databáze
- `DELETE` – zmazanie záznamov v databáze

Žiadosti musia mať v hlavičkách parametre s časovou známkou (`timestamp`), tabuľkou, token, ktorým sa aplikácia vytvárajúca žiadosť identifikuje (`token`) a tajný kľúč jedinečný pre každú aplikáciu (`signing_secret`). Telo žiadosti musí obsahovať kľúč `key` so zašifrovanou hodnotou kľúča, ktorým sa šifroval `payload` a nakoniec samotný `payload`, v ktorom je zašifrované telo žiadosti.

Po prijatí žiadosti sa prv kontroluje formát hlavičky a tela či obsahujú všetky potrebné parametre. Ak sa kontrola vydarila, tak sa ako prvé v metódach dešifruje telo do formátu čistého textu a znovu sa prevádza kontrola dešifrovaného tela či odpovedá formátu JSON a obsahuje potrebné kľúče pre úspešnú komunikáciu s API. V prípade ak aj táto kontrola podarila tak metódy začnú pracovať nad databázou s prijatými údajmi. Pokiaľ pri niektorom z krokov nastala chyba tak notifikačný bot vracia chybovú hlášku spolu s návratovým kódom `500` v opačnom prípade vráti požadované dáta v zašifrovanej podobe s návratovým kódom `200`.

Súčasťou archívu v prílohe je aj pythonovský skript `api_tester.py`, pomocou ktorého je možné aplikačné rozhranie otestovať. Umožňuje si nadefinovať prípadne vložiť vlastné telo vo formáte JSON, nastaviť jednu z REST metód a použiť vlastnú časovú známku v tvare unixového času.

Šifrovanie a dešifrovanie žiadostí

Notifikačný bot šifruje dáta vo formáte čistého textu symetrickou šifrou, ktorej kľúč na druhú stranu šifruje asymetrickou šifrou. Do tela odpovede vloží zašifrované dáta pod kľúč `payload` a zašifrovaný kľúč, ktorým sa šifrovali dáta pod kľúč `key`. Hlavičky obsahujú `token` z prijatej žiadosti, na ktorú bot odpovedá, podpis tela žiadosti a časovú známku, `timestamp`, ktorá bola použitá pri vytvorení podpisu. Podpis sa robí podobne ako v sekcií Podpis žiadosti. Z tabuľky `accounts` si bot načíta tajný kľúč podľa tokenu v hlavičkách žiadosti a vytvorí podpis pomocou základného reťazca spojeného s časovou známkou a zašifrovaným telom správy. Výsledný reťazec, ktorý vznikne znovu spojením základného reťazca s vytvoreným podpisom, vloží do

hlavičiek pod kľúč `signing_secret`. Na zašifrovanie dát sa používa Fernet, čo je systém pre symetrické šifrovanie a dešifrovanie zahrnutý v pythonovskej knižnici `cryptography`.

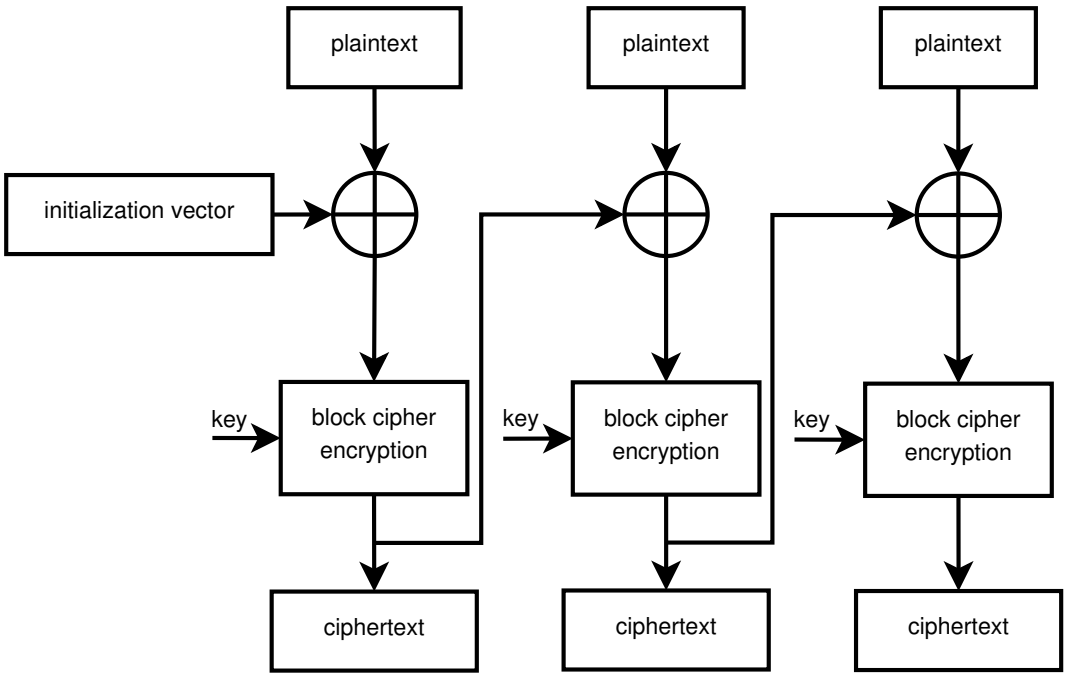
Výpis 2.4: `Príklad generovania kľúča, šifrovania a dešifrovania správy`

```
from cryptography.fernet import Fernet

key = Fernet.generate_key()
f = Fernet(key)

ciphertext = f.encrypt(b"Plaintext")

f = Fernet(b"secret_key")
plaintext = f.decrypt(ciphertext)
```

Fernet šifruje pomocou algoritmu *Advanced Encryption Standard – štandard pre pokročilé šifrovanie* (AES), ktorý šifruje/dešifruje dáta rozdelené do rovnakých pevných blokov s rovnakým kľúčom[30] v móde *Cipher Block Chaining – reťazenie šifrových blokov* (CBC) pomocou vypĺňania *Public Key Cryptography Standards – skupina štandardov pre kryptografiu s verejnými kľúčmi* (PKCS)**#7** [31]. Pri CBC móde sa pred šifrovaním blok xoruje s predchádzajúcim zašifrovaným blokom kde prvý blok je xorovaný s inicializačným vektorom. Zreťazené bloky sú závislé na všetkých predchádzajúcich blokoch. Pri poškodení bloku nie je možné dešifrovať ďalšie nasledujúce bloky. Diagram šifrovania je naznačený na obrázku Obr. 2.7. Dešifrovanie prebieha rovnakým štýlom kde po dešifrovaní kľúčom sa daný blok xoruje s predchádzajúcim zašifrovaným blokom. Prvý dešifrovaný blok sa xoruje s inicializačným vektorom[32]. Vypĺňaním sa docieli toho aby každý blok v móde CBC mal vždy rovnakú dĺžku bitov v prípade krátkej bitovej dĺžky čistého textu.[33].

Na zašifrovanie kľúča sa používa algoritmus *Rivest, Shamir, Adleman* (RSA). Jedná sa o asymetrickú šifru založenú na súkromných a verejných kľúčoch. Verejný kľúč môže byť verejne zdieľaný. Používa sa na šifrovanie správ posielaných notifikačnému botovi. Na druhú stranu súkromný kľúč musí byť uchovaný na bezpečnom mieste vlastníkom. Slúži na dešifrovanie správ. Ako bot tak aj užívateľ musia mať vlastné súkromné kľúče, ktorými dešifrujú správy, ktoré im smerujú a musia mať medzi sebou vymenené svoje verejné kľúče pre šifrovanie správ[34].

Na generovanie súkromného a verejného kľúča sa použil nástroj príkazového riadka `openssl`, ktorý používa rôzne kryptografické funkcie shellovskej krypto knižnice OpenSSL. Generovanie súkromného kľúča sa vykoná pomocou parametra `genrsa`,



Obr. 2.7: Príklad šifrovania v režime CBC.

možnosti `out` s parametrom výstupného súboru a nakoniec špecifikovaním dĺžky kľúča[35]. Pri verejnom kľúči sa najprv načíta súkromný kľúč s argumentom `rsa` a možnosťou `in` s parametrom cesty k súkromnému kľúču. Špecifikuje sa názov verejného kľúča s možnosťou `out` a parametrom názvu. Predvolene sa na výstup dáva privátny kľúč. S možnosťou `pubout` sa docieli toho aby bol na výstupe verejný kľúč. Nakoniec sa v parametre možnosti `outform` špecifikuje formát výstupu verejného kľúča[36]. Kľúče sa ukladajú s príponou *Privacy Enhanced Mail* (PEM) a sú kódované vo formáte tlačiteľných znakov[37].

```
openssl genrsa -out private_key.pem 2048

openssl rsa -in private_key.pem \
        -out public_key.pem \
        -pubout \
        -outform PEM
```

Na konci sa zašifrované telo zakóduje pomocou dátového formátu Base64 na reťazec obsahujúci tlačitelné znaky a uloží sa do tela odpovedi pod kľúč `key`.

Výpis 2.5: Kódovanie a dekódovanie do/z dátového formátu Base64

```
encoded_payload = base64.b64encode(b'ciphertext')

decoded_payload = base64.b64decode(encoded_payload)
```

Dešifrovanie prebieha rovnakým spôsobom len z opačného smeru. Po úspešnej kontrole, že formát žiadosti je správny, bot dekóduje hodnotu kľúča `key` z tela žiadosti do binárnej podoby a prostredníctvom svojho súkromného kľúča dešifruje kľúč na dešifrovanie hodnoty kľúča `payload` z tela žiadosti. Týmto krokom notifikačný bot získal rozšifrované dáta, ktoré ďalej spracováva vo svojich metódach.

Výpis 2.6: Príklad asymetrického šifrovania a dešifrovania správy

```
with open("public_key.pem", "rb") as key_file:
    public_key = load_pem_public_key(key_file.read())

message = b"plaintext"
ciphertext = public_key.encrypt(
    message,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)



with open("private_key.pem", "rb") as key_file:
    private_key = serialization.load_pem_private_key(
    key_file.read(),
    password=None,
)
plaintext = private_key.decrypt(
    ciphertext,
    padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
    )
)
```

Formát dát žiadostí

GET

Metóda má povinný kľúč `table`, ktorý obsahuje názov tabuľky, z ktorej bude vracia údaje a minimálne jeden kľúč, ktorý špecifikuje meno stĺpca danej tabuľky. Získanie všetkých užívateľov a kanálov, ktorý odoberajú monitorovanú stránku s identifikačným číslom 1:

```
{"table":"subscriptions", "site_id":"1"}
```

POST

Metóda sa delí na dve časti. V prvej časti metóda má povinný kľúč `table`, ktorý obsahuje názov tabuľky, do ktorej sa bude vkladať záznam. Pre úspešné vloženie potrebuje mať v kľúčoch nastavené všetky nenulové stĺpce. Pridanie nového záznamu medzi monitorované webové stránky:

```
{
    "table":"site_list",
    "name":"VUT: Korona opatrenia",
    "url":"https://www.vutbr.cz/koronavirus"
}
```

Druhá časť sa venuje posielaniu správ užívateľom a kanálom na Slack. Metóda obsahuje dve povinné kľúče. Prvý kľúč `message` obsahuje kontext posielanej správy a druhý kľúč `send_to_slack_channel` pozostáva z názvu alebo ID kanálu.

```
{
    "message": "Testovanie API.",
    "send_to_slack_channel": "D0123456789"
}
```

PUT

Metóda má povinný kľúč `table`, ktorý obsahuje názov tabuľky, v ktorej sa budú aktualizovať údaje a dva kľúče `where` a `set` obsahujúce slovníky špecifikujúce aktualizované stĺpce a aké stĺpce sa budú meniť. Zmena všetkých záznamov so `site_id` rovným 1 na 2:

```
{
    "table":"subscriptions",
    "where": {"site_id":"1"},
    "set":{"site_id":"2"}
}
```

DELETE

Metóda má povinný kľúč `table`, ktorý obsahuje názov tabuľky, z ktorej bude mazať údaje a minimálne jeden kľúč, ktorý špecifikuje meno stĺpca danej tabuľky. Zmazanie monitorovanej webovej stránky s identifikačným číslom 1:

```
{"table":"site_list", "id":"1"}
```

2.4 Monitorovanie webových stránok

Princím monitorovania webových stránok spočíva vo volaní individuálnych monitorov v slučke, ktoré sa nachádzajú v adresári `sites`. Každý monitor musí vracať inicializovanú triedu `Site`, ktorú naplní s najnovším článkom monitorovanej stránky. Po vrátení sa nasledovne kontroluje daný artikel voči databáze na možnú zhodu. V prípade ak zhoda nenastala sa vytvorí nový záznam s daným artiklom a pripraví sa žiadosť na poslanie notifikačnému botovi. Vytvorenie podpisu prebieha rovnakým spôsobom ako v sekcií 2.3.8, len s tým rozdielom, že sa nečíta z databáze ale použije sa pridelený tajný kód. Ako metóda žiadosti sa používa HTTP POST metóda. Telo žiadosti obsahuje slovník s kľúčmi `title`, `article`, `link` a `site_id`, ktoré už ako z ich názvy vyplýva obsahujú vlastnosti najnovšieho článku spolu s číselným identifikátorom daného monitora. V hlavičkách sa nachádza `token` na identifikovanie žiadostí, že sa jedná o žiadosti pochádzajúce z monitorov stránok, `signing_secret` obsahujúci podpis tela žiadosti a `timestamp` ako to časovú známku, pomocou ktorej sa podpis generoval. Hlavný program pošle tri žiadosti s časovým rozdielom 300 milisekúnd v prípade ak notifikačný bot nebude dostupný predchádzajúcou žiadosťou. V prípade troch nepodarených pokusov končí svoju snahu o spojenie a pokračuje ďalej v procese. Po spustení všetkých monitorov na konci slučky, pred uspaním hlavného programu, maže z databázy záznamy staršie ako 30 dní.

2.4.1 Premenné prostredia

Na správny chod monitorov webových stránok musia byť nastavené povinné hodnoty v premenných prostredia. Hodnoty sa nastavujú v `Dockerfile` prípadne `docker-compose.yml`, ktoré sú priložené v prílohe. Hlavný program k povinným premenným pristupuje pomocou metódy `GET` triedy `ScraperAuth` z adresára `scraper_config`.

Povinné premenné prostredia

- `WEBSCRAPERS_TOKEN` – obsahuje priradený token uložený v databáze `auth.db`
- `WEBSCRAPERS_SERVER_LISTENER` – URL adresa, na ktorej načúva bot
- `WEBSCRAPERS_SECRET` – jedinečný tajný kľúč na vytvorenie/kontrolu podpisov

Nepovinné premenné prostredia

- `WEBSCRAPERS_DELAY_IN_MINUTES` – oneskorenie v minútach ako často sa kontrolujú webové stránky [10]

2.4.2 Trieda Site

Trieda je zadefinovaná v súbore `site_class.py`. Obsahuje štyri premenné, ktoré sa získavajú a inicializujú pomocou metód `get` a `set`.

- `__title` – názov nového článku
- `__article` – obsah nového článku
- `__url` – URL nového článku
- `__scraper_id` – jedinečný číselný identifikátor monitora

2.4.3 Logovanie

Hlavný program monitorov webových stránok loguje svoje správy priamo na výstup do konzoly, ktoré majú level dôležitosti INFO a vyššie. Správy všetkých levelov taktiež ukladá do súboru `logs/scraper.log`. Samotná konfigurácia logera monitorov sa nachádza v súbore `scraper_logger.py`.

2.5 Docker

Súčasťou prílohy sú aj prislušné, `Dockerfile`, súbory pre obe skripty. Súbory obsahujú inštaláciu potrebných balíkov pre chod skriptov, prípravu stromovej štruktúry a nastavené premenné prostredia. Build, inštalácia obrazov, zdieľanie adresárov, nadviazanie sieťového spojenia medzi kontajnermi, zdieľanie hosťovských portov a samotné nasadenie sa nastavuje v konfiguračnom súbore `docker_compose.yml`. Spustením príkazu sa zadefinované služby vytvoria a spustia v tzv. halde.

```
docker-compose up -d
```

Záver

Cieľom bakalárskej práce bolo preskúmať a vybrať najvhodnejšiu možnosť ako notifikovať užívateľa. Implementovať modul v programovacom jazyku Python, pomocou ktorého je možná základná komunikácia na zvolenej komunikačnej platforme. V tejto práci bola vybraná komunikačná platforma Slack pretože kontinuálne inovuje svoje API, obohacuje ho o stále nové a nové funkcie, a má rozsiahlo zdokumentovanú dokumenáciu s aktívnou komunitou ľudí.

Práca obsahuje dva hlavné skripty programov. Samostatný notifikačný bot a správcu monitorov webových stránok. Bot využíva vlastný komunikačný modul ako sprostredkovateľa s komunikačnou platformou Slack. Ďalej sú implementované funkcie na odosielanie správ užívateľom, reagovanie na správy od užívateľov pomocou odpovede na dané správy alebo spustením ďalších udalostí, správa odberov s najnovšími článkami z monitorovaných webových stránok a tak isto zabezpečené aplikačné programové rozhranie, cez ktoré aplikácie tretích strán sú schopné s botom komunikovať pomocou REST metód. Monitorovanie webových stránok je rozdelené do viacerých skriptov pre jednoduché pridávanie alebo odoberanie webových monitorov.

Práca bola testovaná na najnovšej verzií programovacieho jazyka Python (3.10). Modul taktiež rieši ukladanie overovacieho Slack API OAuth Tokenu, predvoleného textového kanála pracoviska a tajného kľúča k overeniu žiadostí. Taktiež rozširuje pythonovskú knižnicu o nové funkcie pre jednoduchšie volanie metód pri posielaní správ alebo súborov na komunikačný kanál.

Na zabezpečenie komunikačného kanála sú použité moderné kryptografické systémy. Požadované funkcie na šifrovanie sú rýchlosť a bezpečnosť, preto nie použitá jediná šifra na zašifrovanie prenášajúcich dát. Symetrická šifra je zvolená na zašifrovanie dát kvôli jej rýchlosti ale na druhú stranu nie je až tak bezpečná, tak preto je použitá asymetrická šifra na zašifrovanie kľúča symetrickej šifry. Zašifrovaný kľúč je pred odoslaním zakódovaný z binárnej podoby na tlačiteľné znaky. Dešifrovanie prebieha rovnakým štýlom, len s rozdielom, že pri dešifrovaní symetrického kľúča sa použije súkromný kľúč príjemcu.

Monitorovanie webových stránok je realizované volaním jednotlivých monitorov v slučke hlavného programu. Po kontrole získaného najnovšieho článku stránky je bot notifikovaný prostredníctvom súkromného rozhrania, ktorý ďalej spravuje rozposielanie správ odberateľom.

Výsledkom bakalárskej práce sú funkčné programy, ktoré riešia všetky body zadania danej práce.

Literatúra

[1] MARLINSPIKE, M. *WhatsApp's Signal Protocol integration is now complete* [online]. 2016, [cit. 01. 12. 2020]. Dostupné z URL: <https://signal.org/blog/whatsapp-complete/>

[2] Facebook. *Overview* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://developers.facebook.com/docs/whatsapp/overview>

[3] Facebook. *Create a WhatsApp Business Account for the WhatsApp Business API* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.facebook.com/business/help/2087193751603668>

[4] Facebook. *List Price Schedule* [online]. 2019, [cit. 01. 12. 2020]. Dostupné z URL: <https://developers.facebook.com/docs/whatsapp/pricing/>

[5] CLEMENT, J. *Most popular global mobile messenger apps as of October 2020, based on number of monthly active users* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/>

[6] Slack Technologies, Inc. *Create a user group* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://slack.com/intl/en-cz/help/articles/212906697-Create-a-user-group>

[7] Slack Technologies, Inc. *Productivity* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://slack.com/intl/en-cz/apps/category/At0EFXUU6N-productivity>

[8] MILLER, R. *Slack hands over control of encryption keys to regulated customers* [online]. 2019, [cit. 01. 12. 2020]. Dostupné z URL: <https://techcrunch.com/2019/03/18/slack-hands-over-encryption-keys-to-regul?guccounter=1>

[9] Slack Technologies, Inc. *ADFS single sign-on* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://slack.com/intl/en-cz/help/articles/230902028-ADFS-single-sign-on>

[10] Slack Technologies, Inc. *Slack plans and features* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://slack.com/intl/en-cz/help/articles/115003205446-Slack-plans-and-features->

[11] Mattermost, Inc *Mattermost vs. Slack for Enterprise* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://web.archive.org/web/20200512185839/https://mattermost.com/mattermost-vs-slack/>

[12] Mattermost, Inc *Secure Messaging with Encryption - Mattermost On-Prem Secure Chat* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://mattermost.com/security/>

[13] Mattermost, Inc *Mattermost Pricing* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://mattermost.com/pricing-self-managed/>

[14] Viber Media S.à r.l. *Features | Viber* [online]. 2019, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.viber.com/en/features/>

[15] Viber Media S.à r.l. *VIBER ENCRYPTION EVERVIEW* [online]. 2019, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.viber.com/app/uploads/viber-encryption-overview.pdf>

[16] CLEMENT, J. *Leading communication apps in the Google Play Store worldwide in September 2020, by number of downloads* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.statista.com/statistics/690864/leading-google-play-communication-apps-worldwide-downloads/>

[17] Telegram. *Telegram FAQ* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://telegram.org/faq>

[18] Telegram. *Telegram Applications* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://telegram.org/apps>

[19] Telegram. *Telegram FAQ/ Q: Why not just make all chats 'secret'?* [online]. 2020, [cit. 01. 12. 2020]. Dostupné z URL: <https://telegram.org/faq#q-why-not-just-make-all-chats-39secret-39>

[20] LINE Engineer. *New generation of safe messaging: "Letter Sealing"* [online]. 2015, [cit. 01. 12. 2020]. Dostupné z URL: <https://web.archive.org/web/20160708212507/http://developers.linecorp.com/blog/?p=3679>

[21] BOBROV, L. H. *Mobile Messaging App Map – February 2018* [online]. 2018, [cit. 01. 12. 2020]. Dostupné z URL: <https://www.similarweb.com/corp/blog/mobile-messaging-app-map-2018/>

[22] Slack Technologies, Inc. *conversations.list* [online]. 2020, [cit. 13. 03. 2021]. Dostupné z URL: <https://api.slack.com/methods/conversations.list>

[23] Slack Technologies, Inc. *chat.postMessage* [online]. 2020, [cit. 13. 03. 2021]. Dostupné z URL: <https://api.slack.com/methods/chat.postMessage>

[24] Slack Technologies, Inc. *Using the Slack Events API* [online]. 2020, [cit. 13. 03. 2021]. Dostupné z URL: <https://api.slack.com/apis/connections/events-api>

[25] Slack Technologies, Inc. *url_verification event* [online]. 2020, [cit. 13. 03. 2021]. Dostupné z URL: <https://api.slack.com/events/url_verification>

[26] Slack Technologies, Inc. *Python slackclient* [online]. 2020, [cit. 16. 03. 2021]. Dostupné z URL: <https://pypi.org/project/slackclient/>

[27] Slack Technologies, Inc. *files.upload* [online]. 2020, [cit. 13. 03. 2021]. Dostupné z URL: <https://api.slack.com/methods/files.upload>

[28] Slack Technologies, Inc. *Verifying requests from Slack* [online]. 2020, [cit. 16. 03. 2021]. Dostupné z URL: <https://api.slack.com/authentication/verifying-requests-from-slack>

[29] HIPP, D. R. *About SQLite* [online]. 2020, [cit. 16. 03. 2021]. Dostupné z URL: <https://www.sqlite.org/about.html>

[30] National Institute of Standards and Technology *Announcing the ADVANCED ENCRYPTION STANDARD (AES)* [online]. 2001, [cit. 12. 04. 2021]. Dostupné z URL: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>

[31] Read the Docs, Inc & contributors *Fernet (symmetric encryption)* [online]. 2020, [cit. 12. 04. 2021]. Dostupné z URL:
<https://cryptography.io/en/latest/fernet/>

[32] DWORKIN, M. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques* [online]. 2001, [cit. 24. 05. 2021]. Dostupné z URL:
<https://csrc.nist.gov/publications/detail/sp/800-38a/final>

[33] HOUSLEY, R. *Cryptographic Message Syntax (CMS)* [online]. 2009, [cit. 24. 05. 2021]. Dostupné z URL:
<https://datatracker.ietf.org/doc/html/rfc5652>

[34] JONSSON, J., KALINSKI, B. *Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1* [online]. 2003, [cit. 24. 05. 2021]. Dostupné z URL:
<https://datatracker.ietf.org/doc/html/rfc3447>

[35] OpenSSL Software Foundation *openssl* [online]. 2021, [cit. 24. 05. 2021]. Dostupné z URL:
<https://www.openssl.org/docs/manmaster/man1/openssl.html>

[36] OpenSSL Software Foundation *rsa* [online]. 2021, [cit. 24. 05. 2021]. Dostupné z URL:
<https://www.openssl.org/docs/man1.0.2/man1/rsa.html>

[37] DigiCert, Inc. *What is PEM Format?* [online]. 2021, [cit. 24. 05. 2021]. Dostupné z URL:
<https://knowledge.digicert.com/quovadis/ssl-certificates/ssl-general-topics/what-is-pem-format.html>

Zoznam symbolov a skratiek

ADFS Active Directory Federation Services – softvérový komponent poskytujúci používateľom SSO

AES Advanced Encryption Standard – štandard pre pokročilé šifrovanie

API Application Programming Interface – programové rozhranie aplikácií

CBC Cipher Block Chaining – reťazenie šifrových blokov

GNU GNU's Not Unix – GNU nie je Unix

HTTP Hypertext Transfer Protocol – internetový protokol určený pre komunikáciu s web servermi

JSON JavaScript Object Notation – zápis dátového formátu nezávislého na počítačovej platforme

LDAP Lightweight Directory Access Protocol – protokol ľahkého prístupu k adresáru

PEM Privacy Enhanced Mail

PKCS Public Key Cryptography Standards – skupina štandardov pre kryptografiu s verejnými kľúčmi

POST metóda požiadavku HTTP, ktorá žiada webový server aby prijal dáta uložené v tele požiadavku

REST REpresentational State Transfer – architektonický štýl na distribuovanie hypermediálneho obsahu

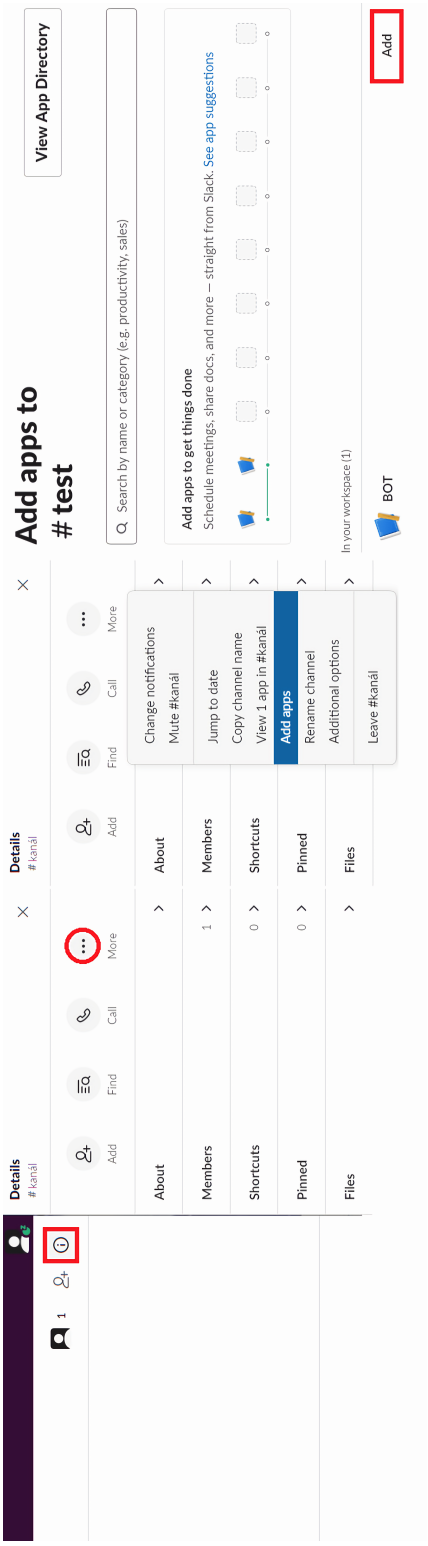
RSA Rivest, Shamir, Adleman

SSO Single sign–on – systém jednotného prihlásenia

URL Uniform Resource Locator – označuje zdroj na internete a protokol, ktorým je možno k zdroju pristupovať

Zoznam príloh

A Obrázky



Obr. A.1: Pridanie aplikácie do komunikačného kanála pracoviska.

B Zdrojové kódy

B.1 Zdrojový kód triedy modulu `notification_bot`

Výpis B.1: `notification_bot/notification_bot_class.py`

```
1  class NotificationBot:
2      __token: str = ""
3      __channel: str = ""
4      __signing_secret: str = ""
5
6      def __init__(self, m_token: str = "invalid_channel",
7              m_channel: str = "invalid_channel",
8              m_signing_secret: str = "invalid_signing_secret"
9      ) -> None:
10         self.token = m_token
11         self.channel = m_channel
12         self.signing_secret = m_signing_secret
13
14     @property
15     def token(cls) -> str:
16         return cls.__token
17     @token.setter
18     def token(cls, m_token: str) -> None:
19         cls.__token = m_token
20
21     @property
22     def channel(cls) -> str:
23         return cls.__channel
24     @channel.setter
25     def channel(cls, m_channel: str) -> None:
26         cls.__channel = m_channel
27
28     @property
29     def signing_secret(cls) -> str:
30         return cls.__signing_secret
31     @signing_secret.setter
32     def signing_secret(cls, m_signing_secret: str) -> None:
33         cls.__signing_secret = m_signing_secret
34
35 notification_bot = NotificationBot()
```

B.2 Príklad uloženia tokenu a kanála do modulu

Výpis B.2: Inicializácia tokenu a kanála v notification_bot module.

```
1 import notification_bot as nb
2
3
4 nb.set_token("xoxb-XXXX")
5 nb.set_channel("maintenance")
```

B.3 Konfiguračný súbor pre tornado server

Výpis B.3: Príklad konfiguračného súboru pre Tornado server.

```
1  import os
2  from dataclasses import dataclass
3
4
5  @dataclass()
6  class ServerConfig:
7      port: int = 4390
8      address: str = '127.0.0.1'
9      token: str = "xoxb-XXXX"
10
11
12 @dataclass()
13 class StaticConfig:
14     img_path: str = "/home/FH/images"
15
16
17 @dataclass()
18 class Config:
19     server: ServerConfig = ServerConfig()
20     server_static: StaticConfig = StaticConfig()
```

B.4 Príklad web servera pomocou webového frameworku Tornado

Výpis B.4: Príklad Tornado web servera

```
from tornado.ioloop import IOLoop
import tornado.web
import tornado.httpserver
from tornado.platform.asyncio import AsyncIOMainLoop
import asyncio
import sys

from config import Config
from handlers import SlackBotHandler

import notification_bot as nb


def make_app() -> tornado.web.Application:
    config: Config = Config()
    app: tornado.web.Application = tornado.web.Application(
        handlers=[
            (r"/command", SlackBotHandler),
        ],
    )

    app.config = config
    nb.set_token(app.config.server.token)
    AsyncIOMainLoop().install()
    return app

if __name__ == "__main__":
    app =  make_app()
    tornado_server = tornado.httpserver.HTTPServer(app)
    tornado_server.bind(app.config.server.port,
        app.config.server.address)
    tornado_server.start()
    asyncio.get_event_loop().run_forever()
```

B.5 Tornado handler

Výpis B.5: Príklad handlera pre tornado server

```
from typing import List
from typing import TYPE_CHECKING
if TYPE_CHECKING:
    from slackBot.config import Config

from tornado.web import RequestHandler
import asyncio
import json
import slack
import sys
import tornado.ioloop
import notification_bot as nb


class SlackBotHandler(RequestHandler):
    async def get(self) -> None:
        pass

    async def post(self) -> str:
        body:     bytes = self.request.body
        request:   dict = json.loads(self.request.body)
        command = request["event"]["text"]
        channel_id = request["event"]["channel"]

        if "ping" in command:
            if await nb.get_channel() == "invalid_channel":
                    await nb.send_message("pong", channel_id)
            else:
                await nb.send_message("pong")
            return "HTTP 200 OK"
        else:
            return self.set_status(400)
```

C Obsah přiloženého archívu

Priložené skripty boli testované vo verzii Python 3.10

- `/` ..... koreňový adresár priloženého archívu
  - `certs` ..... adresár s certifikátmi
  - `databases` ..... adresár s databázami
  - `logs` ..... adresár s logovanými výstupmi
  - `tornado_server` ..... adresár s notifikačným botom
    - `src` ..... adresár so zdrojovými súbormi notifikačného bota
    - `Dockerfile`
    - `requirements.txt` ..... potrebné knižnice notifikačného bota
  - `webscrapers` ..... súbory so skriptami na monitorovanie webových stránok
    - `src` ..... adresár so zdrojovými súbormi monitorov webových stránok
    - `Dockerfile`
    - `requirements.txt` ..... potrebné knižnice pre monitory webových stránok
  - `api_tester.py` ..... skript na testovanie REST API notifikačného bota
  - `api_tester-docs.zip` ..... archív s dokumentáciou API testera
  - `docker-compose.yml`